Weekly Report

January 1, 2018

1 Work

1.1 降维

降维工作的基本已经完成，但是很多细节方面需要打磨。本周主要在代码上添加了对结果的评估函数，以及论文完成了相关工作和部分试验内容。目前还是一些内容的堆砌，好需要好好整理一下。加速方面，KNN使用了蔡登老师他们的方法，大约有4-5倍的加速，我们的方法在小数据集上有8倍的加速，然而数据量大了以后加速会掉到5倍。这是由于K-means迭代次数过多占据过多的时间，可以考虑降低Kmeans的迭代要求，或者采用其他的分割数据的方法。

1.2 工作进度

Table 1: 工作进度

TASK	PROGRESS	DATE
dimension reduction	1)节省内存；2)调整参数；3)增加评估函数	1.10
location2vec专利		
*2Vec survey		1.30

2 Paper Reading

2.1 PixelSNE: Visualizing Fast with Just Enough Precision via Pixel-Aligned Stochastic Neighbor Embedding

文章在tsne投影过程中，考虑到四叉树的构建中最小单元应该大于像素点的大小，这样符合人的认知并且提高了计算效率。

2.2 Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds

经典的locally linear embedding算法，主要思想是一个点可以由他周围点的线形表达构成。我们的目标是使得重构的误差要在低维空间中比较小。

2.3 Laplacian Eigenmaps for Dimensionality Reduction and Data Representation

经典的Laplacian Eigenmaps算法，主要思想是降低高维空间中临近点在低维空间中的距离：$\sum \|y_i - y_j\| W_{ij}$

2.4 Efficient Algorithms for t-distributed Stochastic Neighborhood Embedding

Arxiv上最新出现的一篇文章，使用快速傅立叶变换加速对梯度的计算。

Bare Demo of IEEEtran.cls for IEEE Computer Society Journals

Michael Shell, *Member, IEEE,* John Doe, *Fellow, OSA,* and Jane Doe, *Life Fellow, IEEE*

Abstract—The abstract goes here.

Index Terms—Computer Society, IEEE, IEEEtran, journal, LaTeX, paper, template.

1 Introduction

This

1.1 Background

Representing high-dimensional data via compact vectors in low-dimensional space has been widely studied in several fields of science. For data visualization, embedding high-dimensional data into low-dimensional space such as 2D is an important part of data analysis. Each data object is represent as a object in the low dimensional data in this way. In low-dimensional space, close data objects are tend to share similar attributes and dissimilar objects are far away from each other. Data objects in low-dimensional space can be visualized as scatter plot first. Other visualization approaches such as scalar field and heat map encode the quality or uncertainty of embedding for further exploration. The analysts are able to get an overview of data distribution and generate hypotheses before data process. The applications of dimensional embedding include visualization, anomaly detection ...?

1.2 Relate Work and Challenges

Generally, dimension reduction technologies convert the high-dimensional data set $X = \{x_1, x_2, ..., x_N, x_n \in \mathcal{R}^D\}$ into low-dimensional (two or three) data set $Y = \{y_1, y_2, ..., y_N, y_n \in \mathcal{R}^2\}$ for visual analysis. Over the past decade, a large number of dimensional reduction methods have been proposed, including linear methods and non-linear methods. LINEAR: PCA+MDS; NONLINER: Isomap, LLE, LLP, t-SNE...

Stochadyiv neighbor embedding (SNE) based dimensional reduction methods have achieved outstanding performance. Maaten proposed t-SNE to solve the crowding problem. However, the computational complexity of t-SNE $O(N^2)$ scales with the size of dataset, it is time consuming to visualize larger data sets. In 2014, Mattern employed KNN graph and quadtree to speed up the gradient computation with the complexity of $O(NlogN)$. Recently, largevis optimize the object function with negative sampling method, which reduce the time complexity in terms of the number of data items $O(N)$.

However, the computational complexity of t-sne scales with the size of data set, it is time consuming to visualize very large data sets.

1.3 Our work

In this paper, we develop a new algorithm to speed up the computation of largevis with the complexity of $O(N/K)$.

1.4 Contribution

In summary, our contribution includes:

- We propose a novel dimensional embedding method, which scales linearly with the number of input data.
- We conduct experiments on various real-world datasets. The results show the efficient of our method with a comparable embedding quality.

1.5 Outline

The remainder of the paper is organized as follows. In section 2, we review the related embedding approaches. We describe the basic t-sne model and introduce our method in Section 3. The experiment results are presented in Section 4. Finally, we draw conclusion in Section 5.

2 Related work

Dimensional reduction approaches reduce the number of input variables to speed up the computation or embeded high-dimensional data into low-dimension space for visual exploration and inherent structure reveal. The dimensional reduction approaches are usually divided into two categories: globally embedding methods and locally embedding methods.

For ... methods, principal component analysis (PCA) [8] is the most popular and widely used method, which minimize reconstruction error between high-dimension data and high-dimension data (Linear Dimensionality Reduction: Survey, Insights, and Generalizations): $\|X - M * M^T X\|$, where X is high-dimensional data, M^T is projection matrix,

- *M. Shell was with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332. E-mail: see http://www.michaelshell.org/contact.html*
- *J. Doe and J. Doe are with Anonymous University.*

Manuscript received April 19, 2005; revised August 26, 2015.

$M^T x$ is embedding result and $M * M^T X$ is reconstruction high-dimensional data based on embedding result. Multidimensional scaling (MDS) [11] is another classical dimensional reduction technique which minimize the distance error. In another word, MDS try to keep the distance between any two data when high-dimensional data are embedded into low-dimensional space: $\sum(d(X_i, X_j) - d(M^T X_i, M^T X_j))$. MDS only requires the pairwise distance which is always useful in some situations. In addition, It is proofed that MDS is equivalent to PCA in Euclidean space [3]. Isomap estimate the geodesic distance instead of Eucliden distance to minimize the pairwise distance error. Geodesic distance is more suitable to reflect nonlinear low dimension manifold such as the "swiss roll" data set. The aim of Sammom mapping [15] is to minimize the distance error which is optimized by steepest descent procedure: $\sum(d(X_i, X_j) - d(Y_i, Y_j))/d(X_i, X_j)$. When data has associated class labels, linear discriminant analysis (LDA) is employed to reveal label information. LDA maximize the distance between different clusters and minimize the distance between data point in each cluster. Neural networks such as self-organize map [10] and autoencoder [7] can be employed to reduce the dimension of data.

Most approaches such as PCA and MDS fail to detect nonlinear manifold in high-dimensional space. Locally embedding methods attempt to preserve the local structure: nearby points in high dimensional space remain nearby and similarly co-located with respect to one another in the low dimension space [16]. The basic idea of locally linear embedding (LLE) [16] is reconstructing data points with the linear combination of neighbors in high dimension space and minimizing the reconstruction error in low-dimensional space. Both Laplacian Eigenmap [2] and locality preserving projections (LPP) [5] attempt to minimize the distance between nearby points. LPP assumes that low dimension embeeding is generated by linear transformation. Therefore, LPP is suitable for new test points.

Unlike with previous methods, stochastic neighbor embedding [6] based approaches use probability rather than the distance to measure the similarity among the data points. The goal of these approaches is to minimize the KL distance between two probability distributions on high dimension space and low dimension space. t-distributed stochastic neighbor embedding (SNE) is proposed to solve the crowing problem [12]. t-SNE shows its significant advantages in generating low-dimensional embedding. However, it is time consuming to visualize larger data sets with $O(N^2)$ computational complexity. To solve this issue, Maaten propose Barnes-Hut-SNE (BH-SNE) [**?**], which ignore the edges with small probability, employ k-nearst neighborhood graph to estimate the rest probability and approximate distance in low dimension space by quadtree. Kim et.al introduce a efficient version, called pixel-aligned SNE (PixelSNE) [9]. PixelSne limits quadtree depth based on screen resolution which guarantees the minimum cell size is not smaller than a pixel. Largevis [17] employs a efficient algorithm to construct K-nearest neighbor graph and speed up the optimization by negative sampling [13] and edge sampling [18] techniques.

3 PRELIMINARIES

The family of SNE based approaches usually minimize the distance between two probability distributions on high dimensional space and low dimensional space. The pipeline of existing algorithms is divided into three parts:

- **Input similarity computation**: compute input similarity in high dimensional space;
- **Output similarity computation**: measure output similarity in low dimensional space;
- **Optimization**: minimize the distance between two similarity distributions.

In this section, we first introduce the connection between two existing methods (t-SNE and LargeVis). Then, we propose our efficient approach which speed up LargeVis significantly.

3.1 t-Distribute Stochastic Neighbor Embedding

t-SNE minimize the Kullback-Leibler divergences between two probability distributions of high-dimensional data and low-dimensional data.

Input similarity computation Mathematically, the probabilities between data points in high-dimensional space are defined as follows:

$$p_{j|i} = \frac{exp(-d(x_i, x_j)^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-d(x_i, x_k)^2/2\sigma_i^2)}$$

$$p_{i|i} = 0$$

$$p_{ij} = \frac{p_{j|i} + p_{j|i}}{2N}$$

Output similarity computation A heavy-tailed distribution is employed to measure the probabilities in low-dimensional space:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}} = \frac{(1 + d_{ij}^2)^{-1}}{Z}$$

where Z denotes the normalization term and d_{ij} represents the Euclidean distance between y_i and y_j.

Optimization The low-dimensional representations are learned by minimizing the Kullback-Leibler divergences between two probability:

$$C = KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

The gradient of the of the cost function C with respect to y_i given by

$$\frac{\partial D}{\partial y_i} = \sum_{j \neq i} 4(p_{ij} - q_{ij})(1 + d_{ij}^2)^{-1}(y_i - y_j)$$

Furtherly, BH-SNE split the gradient into two parts:

$$\begin{aligned} \frac{\partial D}{\partial y_i} &= 4(\sum_{j \neq i} p_{ij} q_{ij} Z(y_i - y_j) - \sum_{j \neq i} q_{ij}^2 Z(y_i - y_j)) \\ &= 4(\sum_{j \neq i} F_{ij}^{attr} - \sum_{j \neq i} F_{ij}^{rep}) \\ F_{ij}^{attr} &= \frac{p_{ij}(y_i - y_j)}{1 + d_{ij}^2} \\ F_{ij}^{rep} &= \frac{q_{ij}(y_i - y_j)}{1 + d_{ij}^2} \end{aligned}$$

where F_{ij}^{attr} denotes attractive force between y_i and y_j and F_{ij}^{rep} represents the repulsive force. Note that t-SNE need to compute input similarity p of attractive force in $O(N^2D)$ and output similarity q of repulsive forces with computational complexity of $O(N^2d)$.

3.2 Barnes-Hut-SNE

Input similarity computation According to the grident of t-SNE, computing the input similarities require $O(N^2d)$ which is too expensive. Recent approaches like BH-SNE and LargeVis employee k-nearest neighbor graph to approximating input similarities. The input similarities are redefined as:

$$p_{ij} = \begin{cases} \frac{exp(-d(x_i,x_j)^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-d(x_i,x_k)^2/2\sigma_i^2)} & if j \in N_k(y_i) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$p_{ij} = \frac{p_{j|i} + p_{j|i}}{2N}$$

Optimization BH-SNE only measure attractive force between k-nearest neighbors which reduce the complexity to $O(kN)$.

$$F_{ij}^{attr} = \begin{cases} \frac{p_{ij}(y_i - y_j)}{1+d_{ij}^2} & if j \in N_k(y_i) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Additionally, BH-SNE approximate the repulsive forces in $O(N\log(N))$ by assigning the same distance between points in two cells of quadtree.

$$F_{ij}^{rep} = \begin{cases} \frac{q_{ij}(y_i - y_j)}{1+d_{ij}^2} & if d_{ij}^2 < \theta \\ \frac{q_{i,cell}(y_i - y_j)}{1+d_{i,cell}^2}, j \in cell & \text{otherwise} \end{cases} \quad (3)$$

3.3 LargeVis

Input similarity computation Since constructing exact KNN graph is time consuming, LargeVis propose an efficient approximate k-nearest neighbor graph construction method. First, an initial k-nearest neighbor graph is built based on random projection trees. Then, LargeVis refine the k-nearest neighbor graph by exploring the neighbor's neighbor to improve the accuracy.

Optimization Though the object function of LargeVis is defined on a graph, we discuss the optimization from the perspective of Kullback-Leibler divergence to be consist with the previous method. We can use the Kullback-Leibler divergence as loss function

$$\begin{aligned} C = KL(P||Q) &= \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \\ &= \sum_i \sum_j p_{ij} \log p_{ij} - p_{ij} \log q_{ij} \end{aligned}$$

Note that the first part of this equation is a constant. Therefore, minimizing the Kullback-Leibler divergence is equal to maximizing the following cost function:

$$\begin{aligned} \min \quad C \Leftrightarrow \quad & min \quad \sum_i \sum_j \quad p_{ij} \log p_{ij} - p_{ij} \log q_{ij} \\ \Leftrightarrow \quad & max \quad \sum_i \sum_j \quad p_{ij} \log q_{ij} \end{aligned}$$

Inspired by the negative sampling techniques, LargeVis randomly select two connected points (y and y_j) in KNN graph and sample M negative points (y_{j_k}) for optimization:

$$O = \sum_{i,j \in E} p_{ij} [\log q_{ij} + \sum_{k=1}^{M} \gamma \log(1 - q_{ij_k})]$$

The gradient of these points are given by:

$$\begin{aligned} \frac{\partial O}{\partial y_i} &= -2F_{ij}^{attr} + \sum_{k=1}^{M} \gamma F_{ij_k}^{rep} \\ \frac{\partial O}{\partial y_j} &= 2F_{ij}^{attr} \\ \frac{\partial O}{\partial y_{j_k}} &= -2F_{ij_k}^{rep} \\ F_{ij}^{attr} &= \frac{p_{ij}(y_i - y_j)}{1 + d_{ij}^2} \\ F_{ij_k}^{rep} &= \frac{\gamma(y_i - y_{j_k})}{d_{ij}^2(1 + d_{ij}^2)} \end{aligned}$$

4 Our Method

In this section, we present a accelerated algorithm of LargeVis, which is faster than LargeVis by 5 times. We employ a faster KNN graph construction technique from EFANNA to accelerate the graph construction. In addition, We assign the gradient to one cluster to speed up the calculation of attractive and repulsive forces.

4.1 Input Similarity Computation

Because the similarities between near neighbors are sufficient to preserve the relationship in high dimensional space, the similarities between dissimilar points are nearly infinitesimal [12]. To this end, we can compute the similarities to k-nearest neighbors for each point. In practice, we employee the KNN graph construction part of EFANNA [4] to accelerate the graph construction. The basic idea behind this approaches is "a neighbor of a neighbor is also likely to be a neighbor". .....TODO.....

4.2 Optimization

.....TODO.....nearby data points share similar forces

We assume nearby data points in low dimensional space share similar forces. We can compute the gradient one time and assign the gradient to nearby data points, which accelerate the optimization process significantly. For example, given two points (x_{j1} and x_{j2}) which are neighbors in low-dimensional space. First, y_{j1} and y_{j2} can share the same negative samples y_{j_k}. Therefore, they will have similar repulsive forces. If two point are also close in high-dimensional space. Then, their attractive forces are supposed to be similar due to similar input similarity.

The whole optimization process includes three stages:

- **Generate clusters** At first, all data points are initialized randomly. Similar data will group into small clusters during the early optimization process.
- **Merge clusters** Since we have observed many small clusters, it is more efficient to merge two similar clusters by moving the whole cluster instead of moving

each point. Therefore, we perform k-means to group small clusters. When we compute the gradient for a data object, we assign the gradient to the whole cluster which the data object belongs to. Similar clusters will merge into a large cluster within several steps.

- **Refine layout** Since there are some data points assigning to wrong cluster, we need to refine the layout by placing these data points to a better position. Then, we stop moving clusters and refine the embedding result at the data point level.

4.2.1 Generate clusters

During the early optimization process, the similar data points (with the same labels) are tend to group into small clusters firstly. Then similar small clusters move to each other into large cluster. However, as mentioned in Barnes-Hut-SNE [19], it is much harder for the optimization to find a good embedding in later stages, especially for large-scale data set. In our experiments, we found that merging small clusters with the same labels is time consuming when there is another cluster lies between. The central cluster will push small clusters away since they are not close in the high-dimensional space (see Figure ?). Therefore, the solution to this problem is enlarging the attractive forces between similar points. Previous works such as t-SNE and Barnes-Hut-SNE [19] multiplied p_{ij} by a constant $\alpha(= 4\ or\ 12)$ to attract similar points into one group. Our work employ a similar strategy by decrease γ of repulsive forces in LargeVis. In practice, we fix $\gamma = 1$ for the first 10% optimization process.

4.2.2 Merge clusters

It is more efficient to move small clusters together instead of move data points one by one. Therefore, we perform k-means algorithm to generate K clusters $C_1, C_2, ..., C_K$, where each cluster c_i contains in data points: $C_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$. Then we are able to maximize the object function with K clusters.

We assume that the data points of each cluster are close in high-dimensional space and share similar local structure. Data point in each cluster will share similar gradient. Thus, we compute the gradient dy of all data points for i-th cluster and move the cluster along the average gradient $\sum_{y_j \in C_i} dy_j$:

$$y_i^{new} = y_i^{old} + \alpha \frac{1}{|C_i|} \sum_{y_j \in C_i} dy_j,\ y_i \in C_i$$

We employ stochastic gradient descent to speed up optimization by approximating the average by a gradient at a single data point:

$$y_i^{new} = y_i^{old} + \alpha dy_j,\ y_i \in C_i$$

In practice, we randomly select a data point, compute the gradient and apply the gradient of one data point to the whole cluster. For two clusters (e.g., C_1 and C_2) with the same labels, we need to move at least $min(|C_1|, |C_2|)$ steps to merge two clusters by moving single data points one by one. We need to move once on cluster-scale.

4.2.3 Refine layout

In this stage, we focus on refining the position of a small number of data points which are assign to wrong cluster and move to a inappropriate position. Therefore, we are supposed to move these data points on point-scale. We use original optimization to refine dimensional reduction result.

5 EXPERIMENTS

In this section, we show the quantitative and qualitative embedding result of our method. We show the result of three experiments. In the first experiment, we study the performance of K-nearest neighbor graph construction and verify the importance of K-nearest neighbor graph accuracy on the quality of embedding. In the second experiment, we compare the efficiency and effectiveness of different visualization algorithms. The third experiment compares the visualization result qualitatively. We conducted all experiments on a single desktop PC with Intel Xeon E5 1660 CPU and 16GB memory.

5.1 Data Sets

We performed experiments on a broad range of data sets and compared our method with state-of-the-art methods.

1) The MNIST[1] data set contains 70,000 grey scale handwriting digital images. Each image contains 28 * 28 = 784 pixels and belongs to one of ten digitals from 0 to 9. Each image is treated as a 784 dimension data point.
2) The fashion-MNIST[2] is a data set consisting of 70,000 grey scale images with labels from 10 fashion product categories. It shares the same image size with MNIST datast. Fashion-MNIST data set is introduced to replace the original MNIST dataset which is too easy for currently machine learning model.
3) The CIFAR-10[3] data set includes 60,000 color images with 32 * 32 resolution. All images are labeled with 10 classes, such airplane, bird, etc. There are 6,000 images in each class.
4) The CIFAR-100 data set is similar with CIFAR-10. There are 100 classes with 600 images each.
5) The street view house numbers (SVHN) data set[4] is a real-world house number image data set obtained from Google Street View images. We use all 630,420 color images on character level format where digits are resized to a resolution of 32 * 32 pixels. The SVHN data set is much harder than MNIST to recognize digits. We employ a pretrained deep neural network to preprocess the image and extract better data representation for visualization. The neural network architecture contains eight layers: the first seven are convolution layers (with batch normalization, ReLU activations and max-pooling over 3 * 3 regions) and the last one is a softmax layer. This network achieve ?% error on training data, ?% error on test data and ?% error on extra data. We input an 1024 dimensional data into network and the number neurons in each layers is 3 * 32 * 32, 32 * 30 *

1. http://yann.lecun.com/exdb/mnist/
2. https://github.com/zalandoresearch/fashion-mnist
3. https://www.cs.toronto.edu/ kriz/cifar.html
4. http://ufldl.stanford.edu/housenumbers/

30, 32 * 28 * 28, 64 * 12 * 12, 64 * 10 * 10, 128 * 3 * 3, 256 * 1 * 1 and 10. We extract 256-dimensional output of the last convolution layer with as learned representation.

6) The twitter data set[5] contains 1.2M word vectors pre-trained by Glove [14]. Each word is represented by a 200-dimension vector. The distance between two word vectors is an efficient measurement of semantic similarity.

TABLE 1
Summary of data sets

Data set	Size	Dimension	Class number
MNIST	70,000	784	10
Fashion-MNIST	70,000	784	10
CIFAR-10	60,000	1024	10
CIFAR-100	60,000	1024	100
SVHN	630,420	256	10
Twitter	1,193,514	200	None

5.2 Embedding Result Evaluation

We preform experiments on above data sets and evaluate the performance with the following dimensional reduction algorithms:

- t-SNE [12]: The t-SNE algorithm is the most popular as well as effective method to visualize high-dimension data.
- Barnes-Hut-SNE [19]: The Barnes-Hut-SNE technique accelerate t-SNE from $O(N^2)$ to $O(NlogN)$.
- LargeVis [17]: LargeVis is a large-scale graph visualization method which embedding the kNN graph in high-dimensional space to low-dimensional space.
- Ours: We introduce our method in section ?.

Evaluation. We adopt 1-NN classifier as the embedding quality metric. We compute the nearest neighborhood classification accuracy based on embedding result. The label of data x_i predicted by k-NN classifier is:

$$\overline{y_i} = arg\max_{c} \sum_{x_j \in N_k(x_i)} I(y_j = c)$$

Therefore, the accuracy is defined as:

$$A = \frac{1}{N}\sum_{i=1}^{N} I(y_i = \overline{y_i})$$

where x_j is the nearest neighborhood of x_i, y_i and y_j is the label of x_i and x_j. I is identification function. $I(y_i = y_j)$ = 1 when $y_i = y_j$, else $I(y_i = y_j)$ = 0.

5.3 Running Time Comparison

As shown in Table 2, we report the running time of various algorithms on different datesets. We measure the running time of kNN construction (S1), embedding process (S2) as well as total time (Total).

In all cases, our method achieves significant speed up than other methods. For example, on small data set such as MNIST, LargeVis took about 200 seconds while our methods took 35 seconds. For larger-scale data set such as Twitter, our methods is ? times faster than LargeVis and ? times faster than t-SNE. In summary, our methods is more scalable and efficient than previous methods.

5. https://nlp.stanford.edu/projects/glove/

5.4 KNN Classifier

We compare the embedding quality of embedding algorithm by using kNN classifier as mentioned in section ?. Figure ? shows the result of classification accuracy by kNN classifier with different k.

5.5 Visualization Comparison

We present the visualization result of LargeVis and our method on MNIST, CIFAR-100 and Twitter data sets in Figure ?. Though we reduce the total number of sampling edges to 1/10, our method achieves comparable result on different data sets.

Figure ? shows the two-dimensional embedding result on MNIST data set. The color of each dot represents the corresponding label. Ten categories of data points are well separated. The 1-NN classifier error of low-dimensional data is 5.5%, which is similar to the error in original space.

Figure ? shows the embedding process of LargeVis and ours. First, similar data points converge in very early stage in our method. Second, similar small clusters are merged into big cluster in several steps. However, LargeVis takes about half the time to aggregate similar clusters.

For twitter data set, there is no obvious cluster structure in visualization result (see Figure ?). Some analysis...

6 Conclusion

In this paper, we present a accelerated version of largevis by combining kmeans. The experiments show the efficiency of our method in comparison with tsne and LargeVis.

In the future, we will....

"I always thought something was fundamentally wrong with the universe" [1]

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgments

The authors would like to thank...

References

[1] D. Adams. *The Hitchhiker's Guide to the Galaxy*. San Val, 1995.
[2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.
[3] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. CRC press, 2000.
[4] Cong Fu and Deng Cai. Efanna: An extremely fast approximate nearest neighbor search algorithm based on knn graph. *arXiv preprint arXiv:1609.07228*, 2016.
[5] Xiaofei He and Partha Niyogi. Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160, 2004.
[6] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.

TABLE 2
Running Time

	MNIST			F-MNIST			CIFAR-10			CIFAR-100			SVHN			Twitter		
	S1	S2	T	S1	S2	T	S1	S2	T	S1	S2	T	S1	S2	T	S1	S2	
LargeVis	195.271	404.769		140.540	400.085		225.958	374.678		341.133	375.911		2014.039	1993.339				
Ours	47.513	51.22		42.038	51.219		51.04	48.151		74.65	47.444		378.31	406.288				
Speed Up	4.1X	7.9X		3.34X	7.8X		4.4X	7.8X		4.6X	7.9X		5.3X	4.9X				

TABLE 3
Accuracy

100NN Graph	KNN Construction	Embedding	Kmeans	Total	1NN Classifier Accuracy
MNIST	40s(99.95%)	37s	2s	77s	93.9457%
fashion-MNIST	37s(98.71%)	38s	2s	75s	72.9029%
Twitter	175s(76.80%)	282s	108s	457s	56.1362%

[7] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[8] I. T. Jolliffe. *Principal Component Analysis and Factor Analysis*, pages 115–128. Springer New York, New York, NY, 1986.

[9] Minjeong Kim, Minsuk Choi, Sunwoong Lee, Jian Tang, Haesun Park, and Jaegul Choo. Pixelsne: Visualizing fast with just enough precision via pixel-aligned stochastic neighbor embedding. *arXiv preprint arXiv:1611.02568*, 2016.

[10] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, Sep 1990.

[11] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[12] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[14] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[15] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.

[16] Lawrence K Saul and Sam T Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of machine learning research*, 4(Jun):119–155, 2003.

[17] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web*, pages 287–297. International World Wide Web Conferences Steering Committee, 2016.

[18] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

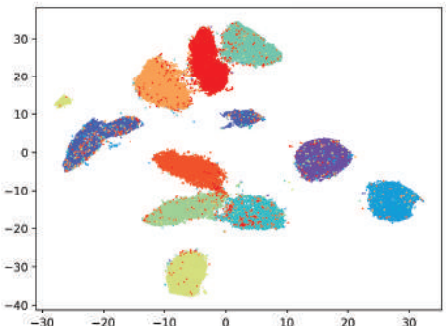
[19] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245, 2014.

PLACE PHOTO HERE

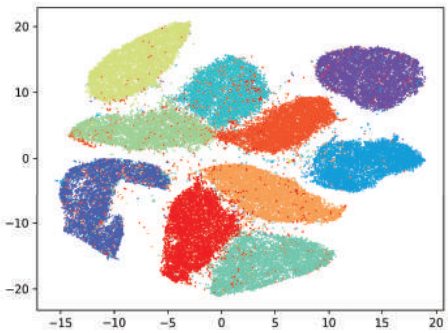
Michael Shell Biography text here.

John Doe Biography text here.

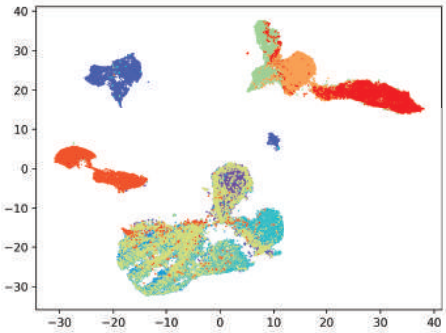
Jane Doe Biography text here.



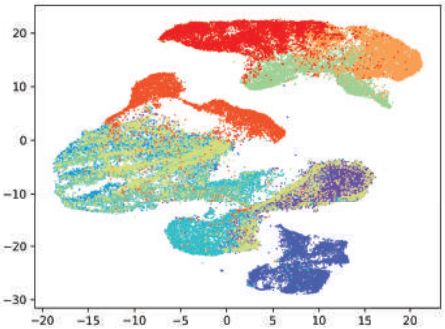
MNIST+LargeVis



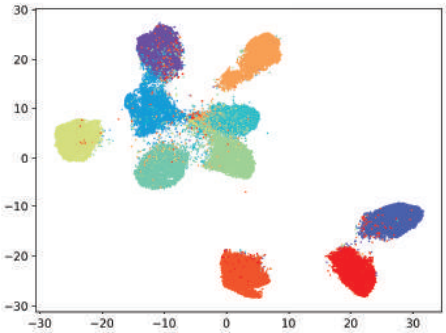
MNIST+Ours



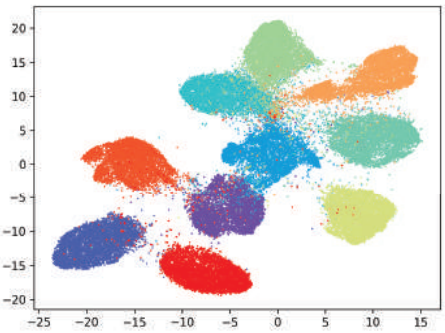
Fashion-MNIST+LargeVis



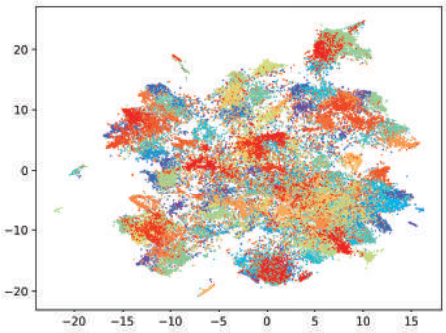
Fashion-MNIST+Ours



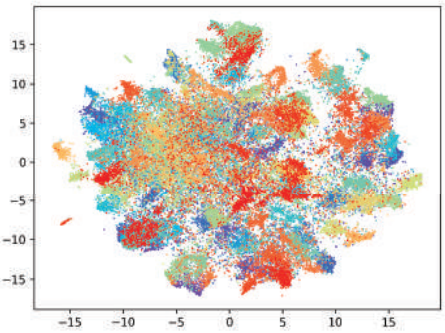
CIFAR 10+LargeVis



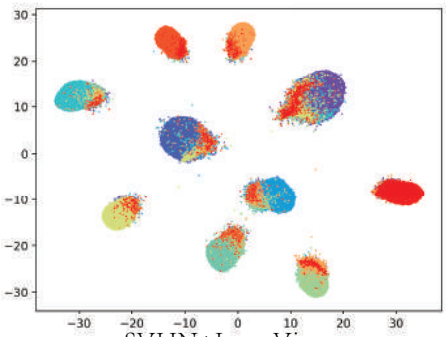
CIFAR 10+Ours



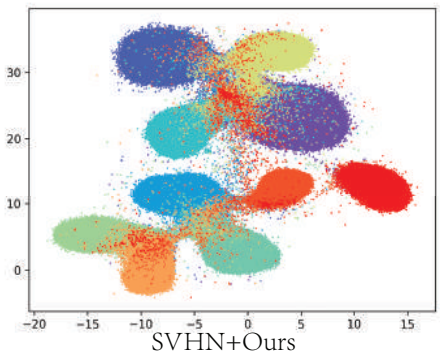
CIFAR 100+LargeVis



CIFAR 100+Ours



SVHN+LargeVis



SVHN+Ours